

AD-A064 590

DEFENSE COMMUNICATIONS ENGINEERING CENTER RESTON VA
SENET SIMULATOR USER'S GUIDE.(U)

F/G 17/2

UNCLASSIFIED

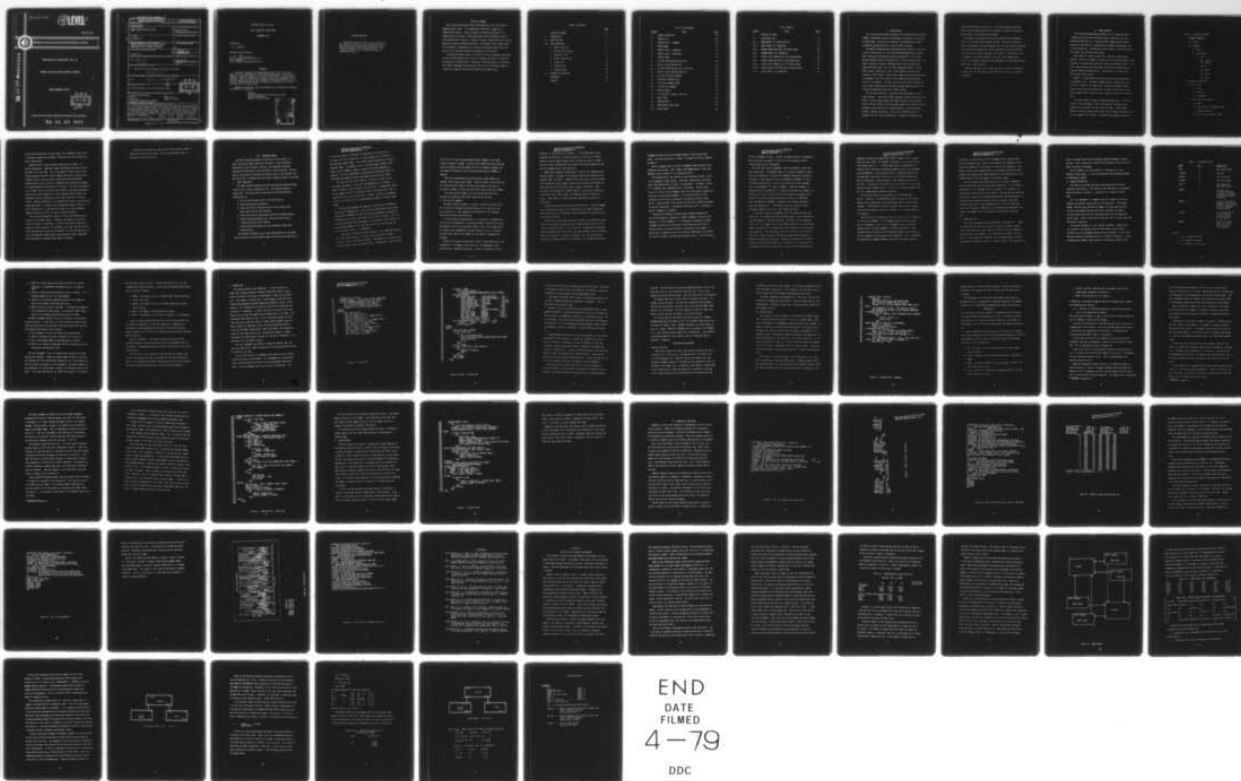
SEP 78 D CASTELLO
DCEC-TN-24-78

SBIE-AD-E100 164

NL

| OF |

AD
A064590



END
DATE
FILMED
4-79

DDC

AD-E 100 164

(12) *na* **LEVEL II**

TN 24-78

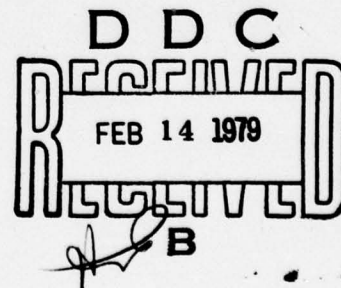


DEFENSE COMMUNICATIONS ENGINEERING CENTER

TECHNICAL NOTE NO. 24-78

SENET SIMULATOR USER'S GUIDE

SEPTEMBER 1978



APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

79 02 07 002

DDC FILE COPY AD A 064590

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 14 DCEC-TN-24-78	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 SENET SIMULATOR USER'S GUIDE.	5. TYPE OF REPORT & PERIOD COVERED 9 Technical Note,	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 D. Castello	8. CONTRACT OR GRANT NUMBER(s)	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Defense Communications Engineering Center Advanced Systems Concepts Branch R740 1860 Wiehle, Ave., Reston, VA 22090	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS N/A	
11. CONTROLLING OFFICE NAME AND ADDRESS (Same as 9)	12. REPORT DATE 11 September 1978	13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) N/A 12 1/4 p.	15. SECURITY CLASS. (of this report) UNCLASSIFIED	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A
16. DISTRIBUTION STATEMENT (of this Report) A. Approved for public release; distribution unlimited. 18 SBIE		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) N/A 19 AD-E100-164		
18. SUPPLEMENTARY NOTES Review relevance 5 years from submission date.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Slotted Envelope Network SOL SENET Speech Activity Simulation Computer Program Integrated Communication Networks Simulation Oriented Language		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This TN classifies the frame allocation techniques which may be used with the circuit-switched and packet-switch integration technique called the Slotted Envelope Network (SENET) concept. It describes a program written in a dialect of the Simulation Oriented Language, SOL-370 Rel 1-79, which simulates a class of the SENET allocation techniques. Example runs of the simulation, statistic, and plot programs are included. A simple three state speech activity model is derived in the appendix.		

DDC
RECEIVED
FEB 14 1978
B

407519

79 02 07 002

TECHNICAL NOTE NO. 24-78

SENET SIMULATOR USER'S GUIDE

SEPTEMBER 1978

Prepared by:

• D. Castello

Approved for Publication:

W L Chadwell

W. L. CHADWELL
Chief, Systems Engineering Division

FOREWORD

The Defense Communications Engineering Center (DCEC) Technical Notes (TN's) are published to inform interested members of the defense community regarding technical activities of the Center, completed and in progress. They are intended to stimulate thinking and encourage information exchange; but they do not represent an approved position or policy of DCEC, and should not be used as authoritative guidance for related planning and/or further action.

Comments or technical inquiries concerning this document are welcome, and should be directed to:

Director
Defense Communications Engineering Center
1860 Wiehle Avenue
Reston, Virginia 22090

ACCESSIBLE TO	
NTIS	White Section <input checked="" type="checkbox"/>
DDC	Gold Section <input type="checkbox"/>
UNCLASSIFIED	<input type="checkbox"/>
RESTRICTED TO	
BY	
GROUP AND/OR INDIVIDUAL CODES	
Dist.	SPECIAL
A	

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to Drs. Vena and Coviello for the original idea and to Mrs. Misch for her tolerance and excellent work during the preparation of the manuscript. Also, I wish to thank all the reviewers for their constructive comments.

EXECUTIVE SUMMARY

One of the architectures being investigated for the future Defense Communications System is an integrated (voice/data), common user, communication system. There are several technical approaches to implementing such a system. These approaches involve different mixes on such network level issues as : analog or digital transmission; and circuit, packet, or hybrid switching technology. The network level issues cannot be satisfactorily addressed until accurate performance models are available for the individual switching subsystems of the network.

Accurate performance models for hybrid switching technology subsystems turn out to be exceptionally difficult (perhaps impossible) to construct on theoretical grounds alone. Therefore, simulation models are required. This report addresses the construction and use of a simulation model for a particular type of hybrid switch known as the SENET switch.

TABLE OF CONTENTS

	<u>Page</u>
EXECUTIVE SUMMARY	iii
I. INTRODUCTION	1
II. SENET SUBTYPES	3
III. SIMULATION MODEL	9
1. SENET Simulation	9
2. The SOL-370 Language	11
3. Simulator Files	16
4. Global Declarations	17
5. Process SOF	23
6. Process Voice	28
7. Process Frame	38
IV. RUNNING THE SIMULATOR	41
REFERENCES	53
APPENDIX	54

LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Title</u>	<u>Page</u>
1.	GLOBAL DECLARATION	19
2.	PROCESS SOF	24
3.	PROCESS VOICE : PREAMBLE	29
4.	SPEECH MODEL	33
5.	PROCESS VOICE : SIMULATOR	34
6.	PROCESS VOICE : CONCLUSION	37
7.	PROCESS FRAME	39
8.	JCL FOR SIMULATOR RUN USING DISK	42
9.	OUTPUT LISTING INITIAL RUN	43
10.	JCL FOR SIMULATOR RUN DISK IN TAPE OUT	44
11.	OUTPUT LISTING CONTINUATION RUN	45
12.	JCL FOR STATISTICS PROGRAM	46
13.	STATISTICS PROGRAM INPUT	47
14.	STATISTICS PROGRAM OUTPUT	48
15.	JCL FOR PLOT PROGRAM	49
16.	PLOTTED RESULTS	51
17.	JCL TRANSLATE, COMPILE, AND LINK	52
A-1	BRADY MODEL	59
A-2	SHERMAN MODEL	62
A-3	PRELIMINARY SPEECH MODEL	64
A-4	SPEECH MODEL	66

LIST OF TABLES

<u>Table</u>	<u>Title</u>	<u>Page</u>
I	SUBTYPES OF SENET	4
II	SIMULATOR FILES	18
A-I	MEASUREMENTS OF SPEECH ACTIVITY	57
A-II	BRADY MODEL EXIT PARAMETERS	60
A-III	MARKOV TRANSITION MATRIX FOR BRADY MODEL	60
A-IV	SHERMAN MODEL EXIT PARAMETERS	61
A-V	STEADY STATE PROBABILITIES FOR SHERMAN MODEL	62
A-VI	MARKOV TRANSITION MATRIX FOR SHERMAN MODEL	62
A-VII	STEADY STATE PROBABILITIES FOR SPEECH MODEL	65
A-VIII	MARKOV TRANSITION MATRIX FOR CONTINUOUS MODEL	66
A-IX	SPEECH MODEL EXIT PARAMETERS	66

I. INTRODUCTION

This note discusses possible types of the Slotted Envelope NETWORK (SENET) transmission concept and describes a simulator for a subclass of these types. By using this simulator the performance of SENET systems may be determined for various traffic loadings.

The Defense Communications Engineering Center (DCEC) is exploring various architectures for the future Defense Communications System (DCS). Among the architectures being investigated is the Slotted Envelope NETWORK (SENET) proposed by Drs. Coviello and Vena [1]. The SENET transmission concept integrates both circuit-switched and packet-switched traffic into a single integrated network. In the SENET concept transmission time is first broken into equal time intervals called frames. Within these frames circuit-switched traffic is assigned to the first region of the frame and packet-switched traffic to the second. The next section of this note discusses the various ways traffic may be allocated to these regions; each of the allocation techniques determines a SENET subtype.

The note then describes a simulator which simulates a class of these subtypes. The class of SENET subtypes includes those which have either a free or upper constrained region for the circuit-switched traffic; which allocate circuit-switched traffic to a fixed slot size; and which, when inactivity is detected within a circuit-switched channel, compress the region. The circuit-switched traffic is assumed to be all voice conversations. Silences in conversational

speech are detected as inactivity. A three state model simulates conversational speech consisting of periods of talking followed by either short or long periods of silence.

The report is divided into four main sections and an appendix. Section II discusses the subtypes of SENET. The actual simulator code is described in the next section, and the final section describes the Job Control Language (JCL) cards required to run the simulator, statistics, and plotting programs. An example is also given of the JCL needed to translate, compile, link, and load another model. Finally, the appendix describes the development of the speech activity model used in the simulator.

The work reported in this Technical Note is part of the author's dissertation for the degree of PhD from the Air Force Institute of Technology.

II. SENET SUBTYPES

The Slotted Envelope Network (SENET) [1] is a technique which integrates real-time data (digitized voice, facsimile, etc) and packet-switched data into a single digital communications network. Basically, the system is a dynamically allocated, synchronous, time division multiplex. Transmission time on links is first partitioned into equal size periods called frames.

Each frame is further divided into a start of frame and two regions. The start of frame is a series of bits used to maintain frame integrity. The remaining two regions contain the network traffic. The first region contains the real-time traffic and the second region contains packet-switched traffic. The method of allocating data within each region varies.

Region I is subdivided into "slots" and each real-time channel is assigned a slot. The SENET concept does not specify that these slots be a fixed or a variable size. Nor does the SENET concept specify that a real-time channel will occupy a single slot or many. The assignment within region II also allows some room for interpretation.

The second region is used for packet-switched data. If there is no data in this category, "idle" characters will be sent on the transmission link until the next start of frame. A data packet usually contains a length count as part of its header; therefore slots are not needed in this region. A problem arises, however, when the

TABLE I. SUBTYPES OF SENET

I. BOUNDARY MOVEMENT

A. Fixed

B. Movement

1. Free

2. Constrained

a. Time

- (1) Fixed
- (2) Varying

b. Limits

- (1) Upper
- (2) Lower
- (3) Both

II. REGION I ALLOCATION

A. Slot Size

- 1. Fixed
- 2. Variable
- 3. Incremental

B. User Activity Detection

- 1. None
- 2. Fill slots with region II data
- 3. Compress region I
- 4. Use for other region I data

space remaining in the region is too small to contain a full packet.

Two choices are available:

- Hold packet until start of region II in next frame.
- Transmit part of the packet this frame and the remainder at the start of the region II in the next frame.

The second method is preferred.

In addition to the allocation strategies within the regions the allocation of the two regions within the frame may vary. If we assume a "boundary" between the two regions then the method of partitioning the frame into the two regions dictates how this boundary moves. In general the boundary may be fixed, freely movable, or constrained.

In the fixed boundary case, a fixed portion of the frame is allocated to region I and a fixed portion to region II. In this case unused bits within the region I must be flagged as idle. The portion of the frame containing these idle bits is effectively lost.

In the free boundary case, channels of the region I type are allocated slots, as needed, up to the entire frame. When no real-time channels are present, there is no region I. Therefore, an unused portion of region I is not lost as in the fixed boundary case.

The constrained boundary falls between the fixed and free cases.

There are various types of constrained boundaries:

- Upper limit,
- Lower limit,
- Range limits.

As their names imply, the constraints may be an upper and/or lower bound on the portion of the frame allocated to one of the regions. In addition, these constraints may be fixed in value or allowed to vary with time (as, for example, dependent on overall network loading).

To summarize, SENET is a time division multiplex for real time and packet-switched data. It handles each type of traffic in one of two regions within its frame. Traffic within region I is assigned space on a slot basis and within region II on a serial basis. The boundary between the regions may be fixed or movable. If it is movable then it may be either free or constrained. For the constrained boundary the limits may be fixed or time varying.

Another dimension of classification is the allocation within region I. Users are allocated channels within region I for as long as the channel is needed. When the user leaves the system, its allocation is placed in a free status. Channels are allocated physical space in region I on a fixed, variable, or incremental basis. In fixed space allocation, slots within the region I are of a predetermined size. Not all slots need be exactly the same size. When a user requests a channel, the region I free slots are scanned until a slot of the required size is found. The variable slot size is the opposite extreme. Slot sizes are not predetermined; rather, they are dynamically determined based on the user requirements. This method becomes unwieldy in practice, especially for users requesting channels with

rates having fractional bits per frame. The incremental slot size is a compromise between the extremes. Slots may be of any multiple of a given minimum size.

Another factor in the allocation techniques for region I is activity detection. Some users need a guaranteed time slot, but may not need it all the time. This is the case for voice circuits with silence detection and for certain command and control circuits which transmit a short "Are you alive?" query once every few seconds. Assuming that lack of activity is detected, the slot may be assigned for other purposes during periods of "silence". The first alternative is to "stuff" the slot with bits which normally would be transmitted during the following region II. (This has been loosely called Time Assignment Digital Interpolation, TADI, when referring to digital voice). Another alternative is to compress the region I portion of the frame when there is silence, and correspondingly expand region II. A third alternative is to overbook the region I, as Time Assignment Speech Interpolation (TASI) is used on submarine cables.

The allocation methods for region I traffic also determine subclasses of the SENET concept. Therefore, the method of assigning slots to channels in the region I portion of the SENET frame may be based on a fixed, varying, or incremental slot size, and the activity on the channels may or may not be detected. If activity detection is used, the detected unused capacity may be used for data, compressed from the region, or used by other region I channels.

Developing one simulation model for all these types of SENET is beyond the scope of this effort. The basic simulation model is described in the next section.

III. SIMULATION MODEL

The SENET simulation program is detailed in this section. In order to help the reader understand the model, a brief functional description of the simulator is given. The Simulation Oriented Language (SOL-370) used to code the model is then described. The next part of the discussion describes the values input to the simulator and the files used. Following this, the actual simulation code is described.

1. SENET SIMULATION

The SENET concept integrates circuit-switched and packet-switched traffic onto a single transmission link. The simulator models a single link. The traffic which loads the link has the following characteristics:

- All circuit-switched traffic is digitized voice.
- Voice activity may be detected.
- Voice calls have a Poisson arrival with an interarrival mean time of VIN ms (milliseconds).
- Voice calls have an exponential hold time of VHOLD seconds.
- Packet-switched messages have a Poisson arrival with an interarrival mean time of DIN ms.
- Packet-switched messages have an exponential length with mean DLEN bits.

The simulator attempts to be a good representation of the SENET link allocation, but many overhead control functions that would exist

in an actual switch are ignored. No overhead bits for the start of frame, call slot reservation, or packet headers are included in the simulated SENET frame. Calls entering the system have no setup or teardown times. The number of bits in each message is added to the queue, but no attempt is made to identify individual messages or packets. All bits transmitted in the packet-switched data region of the frame are removed from the data queue during the next frame.

The simulator does try to emulate realistically data arrival and frame allocation. Voice calls and messages arrive randomly. If there is room for its allocation, the voice call is immediately added to the number of calls in progress, NVOICE; its ending time, TEND, is determined; and its voice activity is sensed. As soon as a message arrives its length in bits is determined and added to the data queue, QUEUE.

The simulator assumes the frame has a time period of TFRAME ms and a size of MSFRAME bits. The circuit-switched region has an upper constraint of MVFRAME bits which must be large enough to allocate up to MNVOICE calls with a fixed slot size of VSLOT bits each. If a call arrives that would cause the number of "in-progress" calls, NOVICE, to be more than MNVOICE calls, the arriving call is lost and the lost calls counter, LVOICE, is incremented.

The frame is allocated by the simulator in the same time sequence as in the SENET concept. The time of the next start of frame, NEXT_SOF, starts the frame allocation. The speech activity during the previous frame of each in-progress call is determined. The allocation

for the call in the circuit-switched region, VFRAME, of the SENET frame is changed if needed. Any calls that ended during the previous frame are removed from the number of calls in progress, NVOICE, and any present allocation in the circuit-switched region, VFRAME, is removed.

Data bits transmitted during the previous frame, NDATA, are removed from the data queue, QUEUE. Then the number of bits which may be transmitted this frame is calculated and added to the region I allocation, VFRAME, to form the total SENET frame allocation, SFRAME.

The above process repeats until the simulation time, TIME, exceeds the simulation time, SIMT, specified by the user.

2. THE SOL-370 LANGUAGE

The SENET simulation model is written in SOL-370 (release 1-79). This simulation language was chosen because of its simplicity and availability. This language is an extension of the language described by Guffee and Ulfers [2].

SOL was originally developed by D. E. Knuth and J. L. McNeley [3] and initially implemented on a Burroughs B-5500. Because the original implementation of SOL was written in ALGOL, which is not supported on the IBM 370, DCA implemented a dialect written in PL/I. To provide the additional capabilities needed for this model, I expanded the language.

Within SOL the basic entity which "flows" in the simulation is the transaction. For example, each voice call is represented in the simulation by a separate transaction. The way a transaction flows in

the model is controlled by a "process." In the SENET model three processes are defined: a control process, a voice call or region I process: and the frame process, which includes the region II model. With this brief introduction let's look at the actual elements of the SOL-370 language used in the simulation.

Communication between transactions in the SOL-370 language occurs through globals. The types of SOL globals used by the simulation are INTEGER, REAL, and TABLE. INTEGER globals are used to store variables which assume only whole numeric values and which must be restored for the simulation run to continue after a break in execution. REAL variables are analogous to INTEGER globals in the restart capability, except REAL globals may contain a fractional part of a numerical value. These types of globals are used completely at the user's discretion.

Variables may also be declared within a process. These are common to a transaction but may not be referenced by any other transaction. When the simulation is restarted, these variables are restored just as are global variables.

In addition to SOL global and local variables, the simulation uses TABLE's. SOL tables are used to write integer values on the log file. They do not store a value. This historical file may then be analyzed and the maximum value of the table, the time average, and the variance may be calculated by a statistics program. Also, two plotting programs are available which use the information in the table declaration

statement and the log file to produce graphs of time versus table value. The value tabulated in a table is recorded by the SOL TABULATE statement.

The SOL language also has control statements that are used by the simulation and include: WAIT, CANCEL, NEW TRANSACTION TO, STOP, AND BREAKOUT. Each of these will be discussed briefly.

At the beginning of the simulation, one transaction starts at the first statement of each process. Additional transactions are stated when a NEW TRANSACTION TO <label> is encountered. The symbol, <label>, is a statement label somewhere within the process. The new transaction will start at the statement label while the present transaction continues with the next statement. All existing values of local variables in the original transaction are copied to the new transaction. These provide initial values for the local variables attached to the new transaction. A transaction continues until a transaction control statement is executed.

Transactions continue to execute each statement sequentially until the transaction encounters a CANCEL statement, the end of the process, or a WAIT statement. If a transaction encounters a CANCEL statement or the end of the process, the transaction dies and all storage used by its local variables is returned to the system. In many cases a transaction must be initiated by some event, for example, the start of frame or the next data message arrival. For this purpose

the WAIT statement is used. The WAIT statement contains an expression which indicates the number of time units the transaction should "sleep" before being "reawakened".

The final two control statements, STOP and BREAKOUT, affect the total simulation. The BREAKOUT saves all variables needed to restart the entire simulation, and the STOP statement terminates the entire simulation. A more general form of the BREAKOUT statement is used in the SENET simulation and is one of the extensions to the language. This is the BREAKOUT TO <label> statement. When the statement is encountered by a transaction, all simulation variables are output; in addition, the label is output as the point where the present transaction will continue execution when the simulation is restarted. After the BREAKOUT statement is executed, the present transaction continues sequentially. The breakout label only applies to the restarted simulation and not to the present execution.

In addition to special statements, SOL has system variables and functions. The predefined system variables used in the simulation are TIME and PRIORITY. The TIME variable is the value of the simulated clock. It is an integer, and for the SENET simulation it indicates the number of simulated milliseconds which have elapsed since the start of the simulation. Zero time always refers to the start of the initial simulation. After a restart, TIME has the same value as when the BREAKOUT statement was executed. Because the simulation is in fact a sequential process and transactions are not truly run in parallel, it is sometime necessary to ensure that, if two or more transactions are

scheduled to wake up the same time, one will always be given precedence over the others. For example, the region I size must be calculated before region II. To ensure that certain transactions are executed first, they may be assigned precedence by giving the system variable PRIORITY a value between 0 and 31. When new transactions are generated, they inherit the priority of the source transaction. When a transaction is being executed, it may change its priority at any time. A transaction with smaller priority value will be executed before one with a higher value. In addition to system variables, SOL also has random sample generating functions.

The functions used by the simulator are the EXPONENTIAL(.) and PR(.) function. The EXPONENTIAL function returns a real valued sample from an exponential distribution whose mean is given by the argument. The PR function returns a logical true value if a sample from a uniform density on the unit interval is less than or equal to its argument.

Because SOL-370 is translated into PL/I, any valid PL/I statement is a valid SOL statement. PL/I statements may be intermixed with SOL-370 statements. Blocks of PL/I statements may be contained between special SOL-370 statements: PLIBEGIN and PLIEND. These statements have no effect on the simulation except to allow the contained statements to be skipped by the translator. This is sometimes an advantage for comment statements which SOL-370 would otherwise

reformat, for large blocks of PL/I statements which take CPU time for the translator scan, and for very complex PL/I statements which exceed the complexity of the parser in the SOL-370 translator. One word of caution: only complete blocks or DO-groups should be placed between the PLIBEGIN and PLIEND statements or the translator will not properly match blocks and groups in the overall model.

Variables may be also declared using PL/I declare statements; such variables are global to the entire simulation. If a variable is declared by a PL/I statement within a process, it is global to all transactions executing within the process. SOL declared variables are saved when a breakout is executed and restored, and the simulation is restarted; PL/I declared variables are not so saved. SOL local variables are local to each transaction; PL/I local variables are local to the process. Therefore, PL/I variables declared within the process are common to all transactions in the process, not just one.

3. SIMULATOR FILES

Nine files are used to run the simulator (see Table II). These files provide for input and output of a user's variables, restart files, and statistics files. Not all of these files are needed for any particular run. Normally, file CARD points to cards in the JCL deck. Files PRINTER and PLIDUMP are set to dummy because no useful information is written on them. All data written to the break and log

files is copied to new files at restart, permitting magnetic tape to be used. After a successful restart and new breakout, the old log and break files may be destroyed.

The JCL needed to run the simulator is discussed in a later section of this report. In the next subsection the simulator program is described in detail.

4. GLOBAL DECLARATIONS

This section discusses the global declaration portion of the simulator (see Figure 1). The simulator uses two types of SOL globals: tables and global variables. In addition, PL/I globals are also used.

The first statement is a comment card which contains within the comment text keywords used by the SOL-370 translator. The keyword \$NUMOC instructs the translator to number the translated card deck. The \$LIST and \$SOURCE keywords cause the translator to produce a printed output containing the translated output and the unmodified source input. These listings are for the user and in no way affect the translator's operation.

The second statement is a PL/I declare statement. Because they are located in the global section of the program, the variables defined within the statement pertain to entire program. In this case, the two variables are defined to be PL/I intrinsic functions. If this statement were removed, there would be no functional change to the

TABLE II. SIMULATOR FILES

<u>NAME</u>	<u>REQ</u>	<u>DESCRIPTION</u>
CARD	A	User input values
SYSPRINT	A	Listings
PRINTER	A	SOL trace table
PLIDUMP	A	PL/I error dumps
\$LOGF	A	Time history of tabulated values
\$BRKOUT	A	A list of all variables required to restart simulation (produced by breakout statement)
\$BRKFIL	R	A list of the values needed to restart and continue the simulation. (copied to \$BRKOUT during restart)
\$OLDLOG	R	The log produced by the previous run (copied to \$LOGF during restart)
NAMEFIL	I	The list of table names for use by statistics and plot programs

NOTES:

A = File is always required.

R = File needed for restart.

I = File needed for initial run.

```

1  /* TALKSPURT MODEL WITH SENET FRAME 8/78 $NUMOC $LIST $SOURCE*/
2      DCL (MIN, CEIL) BUILTIN;
3      INTEGER SIMT, TBREAK, TFRAME, NEXT_SOF, VSLOT, DLEN,
4          LVOICE, NDATA;
5      REAL VIN, VHOLD, DIN, TDATA;
6      TABLE ( 0 BY 1 TO 1000000) TQUEUE,
7          ( 0 BY 1 TO 1000000) TVFRAME,
8          ( 0 BY 1 TO 1000000) TSFRAME,
9          ( 0 BY 1 TO 1000000) TNVOICE;
10     INTEGER MQUEUE, MSFRAME, MNVOICE, MVFRAME ;
11     INTEGER QUEUE, SFRAME, NVOICE, VFRAME ;

```

Figure 1. Global Declaration

program except for a PL/I compiler informative error message. Following this statement, the real "meat" of the global declaration portion of the simulator begins.

The next statement on the following two lines is the first SOL global declaration. The eight variables are defined as full word integers. Because these variables are declared in an INTEGER statement, their values will be saved whenever a breakpoint is encountered during execution of the simulator. When the simulation is restarted, their values will be restored before transaction execution is restarted. Actually, some of these variables are reset by the user immediately after restart. The variables are described in more detail below:

- SIMT is a value input from file CARD each simulation start or restart. It is the time in seconds of the simulator clock when the next breakpoint should be taken and the present simulation halted.
- TBREAK is a value input from file CARD each simulation start or restart. It is the number of seconds between snapshot printouts.
- TFRAME is a value input from file CARD only the first simulation start. It is the time in milliseconds of the simulated SENET frame.
- NEXT_SOF is an internally generated value indicating the time of the next SENET start of frame marker.

- VSLOT is a value input only during the initial run of the simulator. It represents the number of bits in a region I slot.
- DLEN is a value input each simulator start or restart. It is the mean number of bits in a data message.
- LVOICE is an internally generated counter of the number of voice calls blocked since time equal zero.
- NDATA is also an internal variable. It indicates the number of bits allocated for data packets in the previous SENET frame, which will be removed from the data queue this frame.

The REAL statement defines its list of variables to be floating single precision. In this case, all the variables are read-in each time the simulation is initialized or restarted except the last one.

The variables are used for the following:

- VIN is number of voice calls which arrive each second.
- VHOLD is the mean hold time in seconds for a voice call.
- DIN is the average number of data messages per second.
- TDATA is an internally generated variable indicating the time of the next data message arrival.

The next statement (lines six through nine) declares the tables used by the simulator. Tables are global names in which a value may be recorded. The first three tables represent bits in the system, and the last table, the number of voice customers. The three integers in the parenthesis are the minimum, increment, and maximum values of the table. The values specified do not affect the function of the table,

and arbitrary values are used. The only required value is for the increment which must be nonzero. The following simulated SENET quantities are placed in tables:

- VFRAME - the number of bits in present SENET frame occupied by active voice calls
- SFRAME - the number of bits in the SENET frame used by both voice and data
- QUEUE - the number of data message bits queued
- NVOICE - the number of voice calls "off-hook" or "in-progress."

Tables are used so that time histories of the values placed into the tables are generated. After the simulation is completed, an independent program generates the statistics for these variables. Because tables are not limited to a maximum value, additional variables must be defined.

The next statement in the global declaration portion of the simulator defines variables which are used as the maximum value for the tables. These maximum values are read in during the initial run of the simulator.

The last line in this section of code defines the integers used to store the values which will be tabulated in the tables described above. This completes the global declaration portion of the simulator. The three processes are described in the following paragraphs.

5. PROCESS SOF

This process controls the simulation. It sets the start of frame time, produces snapshot listings of the table values, causes restart variables to be saved, and performs all input to the simulation. The process is placed first in the program so that the first transaction generated by the SOL operating system will start in this process. This transaction will also be the first activated when the simulation is restarted. In either case the first activities of the process are to input the appropriate variables and to list them. The transaction then enters a loop. In this loop the transaction calculates the time of the next start of frame, checks snapshot requirements, checks for termination time, and after termination goes to sleep for one SENET frame period. When reactivated, the transaction returns to the start of the loop. Because this process is always activated first and because it checks for the end of simulation, it is referred to as the control process.

The first statement (see Figure 2) names the process, SOF, and specifies that only a single transaction will use the process and that no resources are used.

Lines 15 and 16 are PL/I statements which declare that variables N and A are local to the process. PL/I statements are used because these variables need not be saved and restored between simulation runs. The next statement sets the priority of the process. The

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
12      PROCESS SOF,T= 1,R=0;
13      /* CONTROL PROCESS      INPUTS DATA, STOPS SIMULATION,
14      PRINTS INTERMEDIATE RESULTS, SETS SOF TIME */
15          DCL N FIXED BINARY(15) STATIC INIT(0);
16          DCL A FLOAT(21) BINARY STATIC INIT(0.);
17          PRIORITY = 1; /* DO THIS PROCESS FIRST */
18      READIN:
19      PLIBEGIN;
20          DCL FLOAT BUILTIN;
21          GET FILE (CARD) LIST(A) ;
22          SIMT = CEIL(1000.*A); /* CHANGE TO MS */
23          GET FILE (CARD) LIST(A) ;
24          TBREAK = CEIL(1000.* A ); /* CHANGE TO MS */
25          GET FILE (CARD) LIST(A) SKIP;
26          VIN = 1000. / A; /* CHANGE TO MS */
27          GET FILE (CARD) LIST(A) ;
28          VHOLD = 1000. * A; /* CHANGE TO MS */
29          GET FILE (CARD) LIST(A) SKIP;
30          DIN = 1000. / A; /* CHANGE TO MS */
31          GET FILE (CARD) LIST(DLEN) ;
```

Figure 2. Process SOF

```

32     IF TIME = 0 THEN
33         GET FILE(CARD) SKIP(1)
34         LIST(MQUEUE,MSFRAME,MVFRAME,MNVOICE,V SLOT,TFRAME);
35     PUT FILE(SYSPRINT)
36     EDIT('SIMULATION TIME',.001*FLOAT(SIMT), ' SEC',
37         'PRINT INTERVAL',.001*FLOAT(TBREAK), ' SEC',
38         'VOICE ARRIVAL RATE',1000./VIN, ' PER SEC',
39         'DATA ARRIVAL RATE',1000./DIN, ' PER SEC',
40         'VOICE HOLDING TIME',.001*VHOLD, ' SEC',
41         'DATA MESSAGE LENGTH',DLEN, ' BITS',
42         'VOICE SLOT SIZE',V SLOT, ' BITS',
43         'FRAME TIME',.001*FLOAT(TFRAME), ' SEC',
44         'MAX QUEUE SIZE',MQUEUE, ' BITS',
45         'MAX VOICE FRAME',MVFRAME, ' BITS',
46         'MAX TOTAL FRAME',MSFRAME, ' BITS',
47         'MAX NUMBER OF VOICE',MNVOICE, ' CALLS')
48     (SKIP,(2)(A(22),F(12,3),A(14)));
49     PUT FILE(SYSPRINT) SKIP EDIT(' LVOICE', ' QUE',
50     ' NVOICE', ' SFRAME', ' VFRAME', ' TIME')
51     (X(13),(6)A(10));
52     PLIEND;
53     START:
54         NEXT SOF = TIME + TFRAME;
55         IF N >= TBREAK THEN
56             DO;
57                 PUT FILE(SYSPRINT) SKIP
58                 EDIT(LVOICE,QUEUE,NVOICE,SFRAME,VFRAME,TIME-TFRAME)
59                 (X(10),(6)F(10));
60                 N = 0;
61             END;
62         N = N + TFRAME;
63         IF TIME > SIMT THEN /* END THIS RUN */
64             DO;
65                 BREAKOUT TO READIN;
66                 STOP;
67             END;
68         WAIT TFRAME;
69         GO TO START;
70     END;
71

```

Figure 2 (contd). Process SOF

priorities of the other two processes have higher values. Therefore, if transaction reactivations are scheduled for identical times, the transaction in this process will be reactivated first.

The label on the next line is used to indicate the restart point for this transaction when the simulation is restarted. This is explained in more detail later.

The next line is an instruction to the translator that all text between PLIBEGIN at line 19 and PLIEND at line 52 may be skipped by the translator and immediately copied to the translated output. This action speeds up translation and stops the translator from reformatting the text. All the statements contained between lines 19 and 52 are complete, valid PL/I statements. No SOL statements are mixed in.

These statements between lines 19 and 52 perform all user inputs and outputs of the simulator. Line 20 is a declaration of a PL/I built-in function. Statements on lines 21 through 31 input the variables, read each simulation start or restart, and convert the values to internal units. Lines 32 through 34 input values which are read only once at the beginning of the simulation. These are the maximum values of the SENET frame structure. Lines 35 through 51 list all input variables and write the heading for the snapshots.

The first set of variables are input in units of seconds or arrivals/ second. Variables input in seconds are converted to milliseconds, and arrivals/second are converted to milliseconds between

arrivals. The SKIP option on the GET statements causes a skip to the next card; that is, two values are input per card. Text or card numbers may follow the two inputs. The last input card is an exception.

The fourth input card is only input at simulator time zero. Six integer values are input: the first four establish table maximums, and the last two set the voice slot size in bits and the SENET frame duration in milliseconds. No error checking is done for these variables, and the user must be careful in assigning values.

All values input on the fourth card must be positive integers. MVFRAME represents the number of bits of the SENET frame which may be allocated for region I data. MSFRAME represents the total number of bits in a frame. Therefore, MVFRAME may not be greater than MSFRAME (i.e., the simulator cannot handle TASI transmission). Thus, MNVOICE, the maximum number of voice channels, may not exceed the region I capacity. Therefore,

$$MNVOICE * VSLOT \leq MVFRAME,$$

must be satisfied.

After the variables are input, the transaction enters the loop in lines 53 to 70. This loop will be executed until the logical test at line 64 becomes true. When the logical test become true, the breakout statement causes all variables and the transaction status to be saved in the break file. In addition, label READIN is established as the restart point. When the simulation is restarted, the break file is read and simulation continues with this transaction at the

statement pointed to by label READIN. The breakout statement does not halt the present simulation. The transaction continues execution to line 67 where the STOP statement terminates the simulation.

The other statements are standard PL/I. The first line sets the next start of frame time global which is used to synchronize all the transactions. The next 10 lines form a primitive modulo counter for the snapshot listing.

6. PROCESS VOICE

This process handles the region I allocation of the SENET frame. This is combined with the region II traffic in process FRAME. Each call is handled as an independent transaction within this process. In theory, as each call arrives a new transaction is initiated to await the next call arrival, and this call is added to the number of "in progress" calls. The present transaction executes the speech activity model (see Appendix A) for this call. This is continued until it is time for the call to "hang up". At this time the transaction removes the call from the "in progress" calls and cancels itself. The actual simulation process varies slightly from this in order to decrease CPU time.

This process is the most complex in the simulation. Its discussion is therefore divided into three parts: preamble, speech simulator, and conclusion. In the discussion of the preamble the meaning of the local variables, call arrival and allocation, are detailed. The

```

72     PROCESS VOICE, T=195,R=0;
73     REAL TM;
74     INTEGER TI,STATE,TALK,LTALK,ACTIVE,TEND;
75     DCL T FLOAT(21) BINARY INIT(0) STATIC;
76     PRIORITY = 2; /* DO THIS PROCESS AFTER FRAME START*/
77     START:
78         WAIT EXPONENTIAL (VIN); /* WAIT FOR CALL ARRIVAL */
79         NEW TRANSACTION TO START; /* WAIT FOR NEXT ARRIVAL*/
80     /* PROCESS THIS CALL */
81         IF NVOICE >= MNVOICE THEN /* NO MORE SPACE IN FRAME*/
82             DO;
83                 LVOICE = LVOICE + 1;
84                 GO TO FINIS;
85             END;
86     /*ADD THIS CALL TO TOTAL CALLS IN PROGRESS*/
87         NVOICE = 1 + NVOICE ;
88         TABULATE NVOICE IN TNVOICE;
89     /* INITIALIZE FOR THIS CALL */
90         TALK = 0; /* NO TALKSPURT IS IN SENET FRAME */
91         STATE, ACTIVE = 1; /*SET INITIAL STATE */
92     /* STORE CALL START TIME FLOATED INTO VARIABLE TI */
93         TM = TIME;
94         TEND = EXPONENTIAL(VHOLD) + TIME; /* CALL END TIME */
95     /* WAIT TIL START OF NEXT FRAME TO RUN MODEL */
96         WAIT NEXT_SOF - TIME;

```

Figure 3. Process Voice : Preamble

speech simulator is then briefly discussed. Finally, the region I allocation, the transaction "sleep" period, and call hang up are discussed.

The discussion will start with the preamble (see figure 3). Because each call is handled by a separate transaction, the number assigned to T in the PROCESS statement must satisfy the relation:

$$T < MNVOICE + 1.$$

This allocates a sufficient number of transactions to this process to simulate MNVOICE simultaneous voice calls, a transaction which awaits the next call, and the present transaction. The number of simultaneous calls which may be simulated by the present simulator is 193.

Lines 73 through 75 declare the local variables to this process. The variables declared in the REAL and INTEGER statements are local to each transaction and are used to store the "state" of the speech simulator for this transaction:

- TM - The time of the next state transition in the speech activity model.
- STATE - The next state to be transitioned to in the speech model.
- TALK - Non zero if space is to be allocated in region I of this frame for this call.
- LTALK - Non zero if space were allocated in region I of the last frame for this call.

- ACTIVE - Non zero indicates that the present state of the speech model represents a talkspurt.*
- TEND - The time the call will hang up.

In addition to the above integer variables the process uses a temporary floating point variable:

- T - The time in floating seconds until the next state transition in the speech activity model.

The last declare statement is again a PL/I built-in function specification to suppress a compiler error message.

Line 76 is only executed at time zero and sets the priority of all transactions in this process. The value assigned ensures that region I allocations are complete before the region II allocations are calculated and the SENET frame is constructed.

In this model voice calls are assumed to have Poisson arrivals. Therefore, they have an exponential interarrival distribution of mean VIN. This is simulated by lines 77 through 79.

After a call arrival, the transaction checks for space availability. If space is not available the number of lost calls is incremented and the transaction cancels itself. This is accomplished by the code starting at line 81.

When the transaction reaches line 86, it is known that there is room for the call. The call is added to NVOICE, which indicates the number of calls in progress (line 87), and the speech activity simulator is initialized by lines 90 through 96. The present time is converted

* Defined in Appendix A.

to a floating value and stored in TM at line 93. Line 94 calculates the time for the call to end. Notice that the call holding time is simulated to have an exponential distribution of mean VHOLD. The final statement causes the transaction to pause until the next start of frame before starting the voice activity portion of the process.

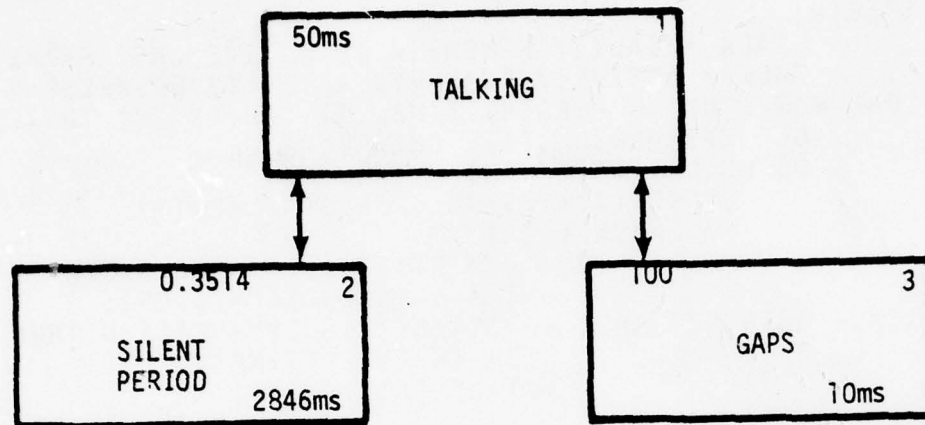
When a transaction reaches this point (line 96) in the process, the initialization or preamble is finished. From here, the transaction starts the voice activity and region I allocation portion of the process.

The speech activity model is shown in Figure 4. The boxes represent the "state" of the speaker. The speaker is uttering sounds in the talking state, state 1. The speaker then transitions to one of the two silent states.* After staying in one of these silent states for a period, the speaker returns to state 1 and the process repeats itself.

In the simulation, the period of time a speaker remains in any state is a sample from a random variable with an exponential distribution. The mean of the distribution is indicated in a corner of the box representing each state. The numbers near the transition lines in the figure indicate the rate at which the given transition will be taken.

In the simulator, the speech activity model starts at line 97 (see Figure 5). The previous state information is saved in lines 101 and 102. The model is run until the the next transition occurs after the present start of frame (the value of TIME).

* Defined in Appendix A.



SPEECH MODEL

Figure 4

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
97 LOOP:
98     IF TIME > TEND THEN /*CALL ENDED*/
99         GO TO HANG_UP;
100  PLIBEGIN;
101      LTALK = TALK; /*REMEMBER IF ACTIVE LAST FRAME */
102      TALK = ACTIVE; /*REMEMBER ACTIVITY OF PRESENT STATE*/
103  /* RUN MODEL UP TO PRESENT TIME. TI (ALIAS TM) INDICATES TIME
104     OF NEXT MODEL STATE CHANGE */
105      DO WHILE (TIME > TM);
106          SELECT (STATE); /* CASE STATEMENT */
107              WHEN ( 1 )
108                  DO; /* STATE 1 HAS TWO EXITS */
109                      T = EXPONENTIAL (0.05);
110  /* TALK GETS SET TO 1 IF STATE 1 IS EVER PASSED THRU RUNNING MODEL
111     UP TO TIME "TIME" */
112                      TALK, ACTIVE = 1;
113  PLIEND;
114                      IF PR(0.0315) THEN /* EXIT TO STATE 2 */
115                          STATE = 2;
116                      ELSE /* EXIT TO STATE 3 */
117                          STATE = 3 ;
118  PLIBEGIN;
119                      END; /* STATE = 1 */
120  WHEN ( 2 ) /* EXIT TO STATE 1 */
121      DO;
122          ACTIVE = 0;
123          STATE = 1;
124          T = EXPONENTIAL (2.8457);
125      END; /* STATE = 2 */
126  WHEN ( 3 ) /* EXIT TO STATE 1 */
127      DO;
128          ACTIVE = 0;
129          T = EXPONENTIAL (.01);
130          STATE = 1;
131      END; /* STATE = 3 */
132  OTHERWISE; /* DO NOTHING */
133  END; /* SELECT */
134  TM = TM + 1000. * T;
135  END; /* TIME > TM */
```

Figure 5. Process Voice : Simulator

The SENET statement in conjunction with the WHEN statements determine which block of code to execute, one block for each state. If the model is in state 1 before the model starts to run, passes through it while running, or ends in it, then a voice slot must be coded in the SENET frame. This is indicated by variable talk being set to "1". This will be checked in the next part of the process. The function of variables T and TM have been explained previously. One additional statement should be explained: line 114.

The talkspurt state* has two exits. The time interval that the process remains in the state has an exponential density. After this interval the talking state will transition to one of the other states. The state transitioned to depends on the rate of transition. Given that the state is left 20 times a second, it transitions to state 3 with probability 19.37/20 and to state 2 otherwise. The notation PR (0.0315) indicates a sample drawn from a uniform density defined on the unit interval. When this sample is less than 0.0315, the next state is chosen to be 2; otherwise 3.

When variable TM becomes greater than the present time, the region I allocation is executed by the transaction. (The simulation time is at a SENET start of frame.) The talkspurt model represents the activity events of this simulated call during the last SENET frame. The simulation is now ready for the region I of the SENET frame to be allocated.

* Defined in Appendix A.

In the conclusion of process VOICE (see Figure 6), this call is allocated in region I. In addition, this section contains the call termination statements plus code to decrease simulation time.

Variable TALK is nonzero if this call needs space allocated in this frame. Variable LTALK indicates whether space was allocated in the previous frame. The allocation for the call need only be changed if it has changed from the previous frame. This is done by lines 138 through 145. After this action, the transaction goes to sleep until the next frame. At this point, we could loop back.

If we loop back to label LOOP after the wait at line 146, the simulation would work properly, but it would use an excessive amount of CPU time. This is because it takes a lot of time to put a transaction to sleep and reawaken it. If we consider the frame size of a typical SENET simulation and the speech model, a way to decrease CPU time becomes apparent. Typically, the SENET frame has a length in the order of 10 ms. The average talkspurt is 50 ms, and the long silence is almost 3 seconds. These two states are occupied over 90% of the time. Therefore, the call occupancy holds constant for many SENET frame periods. The transaction waits a single frame. If there is no state transition scheduled for this frame, the activity of the present state is used to determine the occupancy of the SENET frame until the start of frame following the next transition time.

```

136 /*CHANGE FRAME ONLY IF SPEECH ACTIVITY HAS CHANGED */
137 PLIEND;
138 IF LTALK = TALK THEN
139     DO;
140         IF TALK = 0 THEN /* HAVE GONE SILENT*/
141             VFRAME = VFRAME-VSLOT;
142         ELSE /* HAVE STARTED TALKING*/
143             VFRAME = VFRAME+VSLOT;
144         TABULATE VFRAME IN TVFRAME;
145     END;
146     WAIT TFRAME;
147 /*CODE FROM HERE ON ADDED TO DECREASE SIMULATION TIME */
148 IF TIME > TM THEN /* NO SAVINGS POSSIBLE */
149     GO TO LOOP;
150 IF TIME > TEND THEN
151     GO TO HANG UP;
152 IF TALK = ACTIVE THEN
153     DO;
154         IF ACTIVE = 1 THEN
155             VFRAME = VFRAME+VSLOT;
156         ELSE
157             VFRAME = VFRAME-VSLOT;
158         TABULATE VFRAME IN TVFRAME;
159     END;
160 TALK = ACTIVE;
161 IF TEND < TM THEN /* CALL ENDS BEFORE NEXT STATE CHANGE */
162     DO;
163         WAIT TEND - TIME; /* SLEEP UNTIL CALL ENDS */
164         GO TO HANG_UP;
165     END;
166 ELSE
167     DO;
168         WAIT TM-TIME;
169         WAIT NEXT SOF - TIME;
170         GO TO LOOP;
171     END;
172 HANG_UP:
173     WAIT NEXT SOF - TIME; /* HANGUP IN NEXT FRAME*/
174     NVOICE = NVOICE-1;
175     TABULATE NVOICE IN TNVOICE;
176     IF TALK = 1 THEN /* TALKSPURT IN PROGRESS*/
177         DO;
178             VFRAME = VFRAME-VSLOT;
179             TABULATE VFRAME IN TVFRAME;
180         END;
181 FINIS:
182     END;
183

```

Figure 6. Process Voice : Conclusion

The final portion of the process terminates the call. The present frame allocation is not changed. The transaction waits until the next frame and then removes the call from the number of calls in progress and removes any region I allocation.

All transactions in this process update the region I allocation, VFRAME, before the final SENET frame occupancy is determined in process FRAME.

7. PROCESS FRAME

The last process (see Figure 7) accumulates incoming messages in the queue and adds the region II traffic from this queue to the region I traffic allocated in process VOICE forming the final SENET frame allocation. Both of these functions are performed by a single transaction. The transaction loops in lines 189 through 197 accumulating message bits in the queue for all messages which arrive before the next start of frame and updating the time of next message, TDATA. When the next message is scheduled to arrive after the start of frame, the transaction drops out of the loop and waits for the start of frame. This permits the transactions in the voice process to complete the region I allocation and for a new start of frame time to be calculated.

At this time the transaction removes the bits transmitted in region II during the previous frame, NDATA, from the queue. A new NDATA is calculated for this frame based on the number of bits remaining in the SENET frame for region II traffic and the number queued.

```

184     PROCESS FRAME,T=1,R=0;
185     DCL
186         DELTA FIXED BINARY(31) STATIC INIT(0);
187     PRIORITY = 3; /* ACTIVATED AFTER CLASS 1 FINISHED */
188     TDATA=EXPONENTIAL(DIN); /* TIME OF FIRST MESSAGE */
189     START:
190         IF TDATA < NEXT_SOF THEN
191             DO;
192                 WAIT TDATA - TIME;
193                 QUEUE = QUEUE+EXPONENTIAL(DLEN); /*QUEUE IT*/
194                 TABULATE QUEUE IN TQUEUE;
195                 TDATA = EXPONENTIAL(DIN) + TDATA; /*TDATA OF NEXT MSG*/
196                 GO TO START; /* WAIT FOR NEXT ONE*/
197             END;
198     /*THIS SECTION DETERMINES THE SENET FRAME */
199     WAIT NEXT SOF-TIME; /*WAIT FOR NEXT FRAME */
200     IF NDATA > 0 THEN
201         DO; /*REMOVE DATA PUT IN PREVIOUS FRAME */
202             QUEUE = QUEUE-NDATA;
203             TABULATE QUEUE IN TQUEUE;
204         END;
205     PLIBEGIN;
206     /*CALCULATE SPACE NEEDED OR AVAILABLE THIS FRAME*/
207     NDATA=MIN(QUEUE,MSFRAME-VFRAME);
208     /*CHANGE FROM LAST FRAME */
209     /* VFRAME ----VOICE THIS FRAME */
210     /* SFRAME ----LAST FRAME OCCUPANCY*/
211     /* NDATA ----THIS FRAME DATA*/
212     DELTA = NDATA + VFRAME - SFRAME;
213     PLIEND;
214     IF DELTA = 0 THEN
215         DO;
216             SFRAME = SFRAME+DELTA; /*CHANGE SENET FRAME*/
217             TABULATE SFRAME IN TSFRAME;
218         END;
219     GO TO START;
220     END;
221     END;

```

Figure 7. Process Frame

The change in allocation between this SENET frame and the previous frame is calculated and stored in temporary variable, DELTA. This value, if not zero, is used to update the frame.

Because of the wait until the present start of frame and the low priority assigned to this transaction, the transaction in the first process updates the start of frame. Therefore, when this transaction loops back to label START, TDATA is compared to the next start of frame and the process continues.

IV. RUNNING THE SIMULATOR

Examples of running the simulator are presented in the first part of this section. These are followed by examples for running the statistics and plot programs. The section concludes with an example of translating and compiling the model. After the extended translator is formally released, some of the dataset names may need to be changed.

Due to the large amount of data recorded on the log file, runs for long simulation periods should use magnetic tape. Many times I failed to give enough CPU time for a simulation. Therefore, the job ended without properly closing the files. If the log file were a member of a disk dataset, all results of the simulation would be lost. Using magnetic tape stops this loss. Even if the simulation aborts, the statistics or plot programs are able to recover most of the data.

However, magnetic tapes can be unwieldy both from the user's and the computer operator's viewpoints. Therefore, a compromise is used. The total simulation usually takes two runs: an initialization run to load the SENET frame to near steady state, and then a run using the variables of interest. The author's compromise is to run the initial simulation using all disk files. The following run then uses these disk files as input and produces new files on tape. The tapes are used as input to the statistics program.

The JCL decks for this type of operation are shown in Figures 8 and 10; output listings are shown in Figures 9 and 11. Notice that

```

//R3363SOL JOB (A100,,50,9,220),'CASTELLO',
//  NOTIFY=R3363,MSGCLASS=Q
//GO EXEC PGM=MODEL,REGION=220K,TIME=(0,50),PARM='/O',
// COND=(EVEN)
//STEPLIB DD DSN=R3363.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=(A,U)
//PRINTER DD SYSOUT=(A,U)
//NAMEFIL DD DSN=R3363.SOL.NAMES(MODEL),DISP=SHR
//PLIDUMP DD DUMMY
//$LOGF DD DSN=R3363.SOL.LOG(MODELO),DISP=SHR OUT
//$BRKOUT DD DSN=R3363.SOL.RESTART(MODELO),DISP=SHR OUT
//CARD DD *
1.000, .990 END SIMULATION TIME,PRINT INTERVAL
150.000,300, VOICE ARRIVAL RATE, CALL HOLDING TIME
10.000,1500, DATA ARRIVAL RATE, LENGTH
1000000,15440,15440,193, 80, 10
/*

```

Figure 8. JCL For Simulator Run Using Disk


```

//R3363SOL JOB (A100,,450,9,230),'CASTELLO',
//      NOTIFY=R3363,MSGCLASS=Q
/*SETUP      NEED TWO TAPES RING IN 2893  2814
//GO2 EXEC PGM=MODEL,TIME=(6,50),
// PARM='/1',REGION=230K
//STEPLIB DD DSN=R3363.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=(A,U)
//PRINTER  DD SYSOUT=(A,U)
//$OLDLOG DD DSN=R3363.SOL.LOG(MODELO),DISP=SHR
//$BRKFIL DD DSN=R3363.SOL.RESTART(MODELO),DISP=SHR
//$BRKOUT DD DISP=(NEW,KEEP),DSN=UR3363.REG02,
//      UNIT=TAPE9,DCB=(RECFM=FB,LRECL=133,BLKSIZE=1330),
//      VOL=SER=002814,LABEL=1
//CARD      DD *
30.000, 1., END SIMULATION TIME,PRINT INTERVAL
0.520,300, VOICE ARRIVAL RATE, CALL HOLDING TIME
500.000,1500, DATA ARRIVAL RATE, LENGTH
/*
//PLIDUMP DD DUMMY
//$LOGF DD DISP=(NEW,KEEP),DSN=UR3363.LOG2,
//      UNIT=TAPE62,LABEL=1,VOL=SER=002893
//STAT2 EXEC PGM=SOLSTAT,REGION=120K
//STEPLIB DD DSN=R3363.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=(A,N)
//NAMME DD DSN=R3363.SOL.NAMES(MODEL),DISP=SHR
//LOG DD DSN=UR3363.LOG2,DISP=SHR,UNIT=TAPE62,
//      LABEL=1,VOL=SER=002893
//PLIDUMP DD DUMMY
//SYSIN DD *
5000 50000 10000
TQUEUE
TVFRAME
TSFRAME
TNVOICE
/*

```

Figure 10. JCL For Simulator Run : Disk In, Tape Out

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

SOL RESTART OCCURRED AT	1010				
SIMULATION TIME	30.000	SEC	PRINT INTERVAL	1.000	SEC
VOICE ARRIVAL RATE	0.520	PER SEC	DATA ARRIVAL RATE	500.000	PER SEC
VOICE HOLDING TIME	300.000	SEC	DATA MESSAGE LENGTH	1500.000	BITS
VOICE SLOT SIZE	80.000	BITS	FRAME TIME	0.010	SEC
MAX QUEUE SIZE	1000000.000	BITS	MAX VOICE FRAME	15440.000	BITS
MAX TOTAL FRAME	15440.000	BITS	MAX NUMBER OF VOICE	193.000	CALLS

LVOICE	QUE	NVOICE	SFRAME	VFRAME	TIME
0	114234	170	15440	6560	2000
0	8117	169	7797	3840	3000
0	14331	168	8905	3360	4000
0	14204	168	7343	3920	5000
0	42725	168	15440	3760	6000
0	21069	167	8020	3680	7000
0	20640	168	9389	4000	8000
0	18020	167	15440	4160	9000
0	12582	167	7151	4640	10000
0	7685	166	9413	3040	11000
0	6861	167	9793	3920	12000
0	27539	167	15440	3760	13000
0	22491	167	5636	3120	14000
0	11302	167	11307	3840	15000
0	13168	167	13757	3280	16000
0	16878	167	14922	3920	17000
0	15977	166	13058	3920	18000
0	23934	165	8301	3840	19000
0	41790	165	15440	4800	20000
0	19285	165	14042	4320	21000
0	24535	165	15440	4400	22000
0	16829	164	9688	4320	23000
0	14801	164	14939	3520	24000
0	13617	162	11485	3840	25000
0	11832	161	13408	3440	26000
0	16675	163	15440	3840	27000
0	16218	163	7584	3840	28000
0	34641	162	15440	3600	29000
0	27985	160	15440	3600	30000

SOL CHECKPOINT NO.	2	HAS OCCURRED AT	30010
ENDED AT	30010	WITH COMPL.CODE	0000 1 2

Figure 11. Output Listing, Continuation Run

the PARM option on the EXEC card is set to zero for the initial simulation and to one for the continuation. In general, the value is set to the number of the breakout on the break file at which the simulation is to be restarted. If the simulation were to be continued after the second run, this field would be set to three.

The time parameter is required on the EXEC card for execution time over 5 minutes. The time required to execute the example program was 23 seconds for the initial run and 390 seconds for the continuation. To estimate run times for other cases, assume that run time is proportional to the voice Erlang loading and the data message arrival rate.

The statistics program uses the NAMME file created by the initial simulation and the \$LOGF from any simulation run. The maximum, average, and variance may be calculated for up to four logged variables for any time period on the log. The statistics are printed at a user specified interval. This running average allows the user to ascertain that the simulation has been run long enough for the statistics to be nearing steady state.

The input variables are shown in Figure 12. The statistics will be calculated from "time begin" to "time end." Statistics are printed each "print increment" and at the end of file on the log. The JCL for a statistics run is shown in Figure 13.

A plot program is also available which produces a time plot of up to four logged variables on the CALCOMP flatbed plotter. Its JCL and plot are shown in Figures 14 and 15. The input variables are

Card 1		
Time begin	Time end	Print Increment
Card 2-5		
Statistics Program Input		

Figure 12. Statistics Program Input

```
//R3363STA JOB (A100,,120,9,120),'CASTELLO',
//  NOTIFY=R3363,MSGCLASS=Q
//STAT1 EXEC PGM=SOLSTAT,REGION=120K
//STEPLIB DD DSN=R3363.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=(A,N)
//NAMME DD DSN=R3363.SOL.NAMES(MODEL),DISP=SHR
//LOG DD DSN=UR3363.LOG1,DISP=SHR,UNIT=TAPE62,
//  LABEL=1,VOL=SER=003707
//PLIDUMP DD DUMMY
//SYSIN DD *
5000 50000 1000
TQUEUE
TVFRAME
TSFRAME
TNVOICE
/*
```

Figure 13. JCL For Statistics Program

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDG

NAME	DIMENSION	CAP	INC	MAX
ENTER BEGIN, END, INCR, TIMES				
BEGIN, END, AND INCR, TIMES	5000			50000
ENTER NAME(DIMENSION) AND MAX VALUE				
ENTER NAME(DIMENSION) AND MAX VALUE				
ENTER NAME(DIMENSION) AND MAX VALUE				
ENTER NAME(DIMENSION) AND MAX VALUE				
TYPE-10000001000000000000 NAME - TQUEUL				
TYPE-10000001000000000000 NAME - TVERAME				
TYPE-10000001100000000000 NAME - TISFRAME				
TYPE-10000010000000000000 NAME - INVOICE				
NAME				
TQUEUL	15000			
TVERAME	15000			
TISFRAME	15000			
INVOICE	15000			
NAME				
TQUEUL	25000			
TVERAME	25000			
TISFRAME	25000			
INVOICE	25000			
NAME				
TQUEUL	30007			
TVERAME	30007			
TISFRAME	30007			
INVOICE	30007			

MAX -	MAX -	MAX -	MAXIMUM	DELTA	DELTA	DELTA	AVERAGE	VARIANCE
0	0	0	4.27250E+04	1.00000E+04	1.00000E+04	1.00000E+04	1.31154E+04	4.97748E+07
0	0	0	5.12000E+03	1.00000E+04	1.00000E+04	1.00000E+04	3.77840E+03	1.78085E+05
0	0	0	1.54400E+04	1.00000E+04	1.00000E+04	1.00000E+04	1.13691E+04	1.26698E+07
0	0	0	1.68000E+02	1.00000E+04	1.00000E+04	1.00000E+04	0.00000E+00	0.00000E+00
0	0	0	4.27250E+04	2.00000E+04	2.00000E+04	2.00000E+04	1.34519E+04	5.06061E+07
0	0	0	5.44000E+03	2.00000E+04	2.00000E+04	2.00000E+04	3.94168E+03	2.34021E+05
0	0	0	1.54400E+04	2.00000E+04	2.00000E+04	2.00000E+04	1.16201E+04	1.26555E+07
0	0	0	1.68000E+02	2.00000E+04	2.00000E+04	2.00000E+04	0.00000E+00	0.00000E+00
0	0	0	4.27250E+04	2.50070E+04	2.50070E+04	2.50070E+04	1.32833E+04	5.05584E+07
0	0	0	5.44000E+03	2.50070E+04	2.50070E+04	2.50070E+04	3.85359E+03	2.41128E+05
0	0	0	1.54400E+04	2.50070E+04	2.50070E+04	2.50070E+04	1.14845E+04	1.29690E+07
0	0	0	1.68000E+02	2.50070E+04	2.50070E+04	2.50070E+04	1.65258E+02	4.86719E+00

Figure 14. Statistics Program Output

```

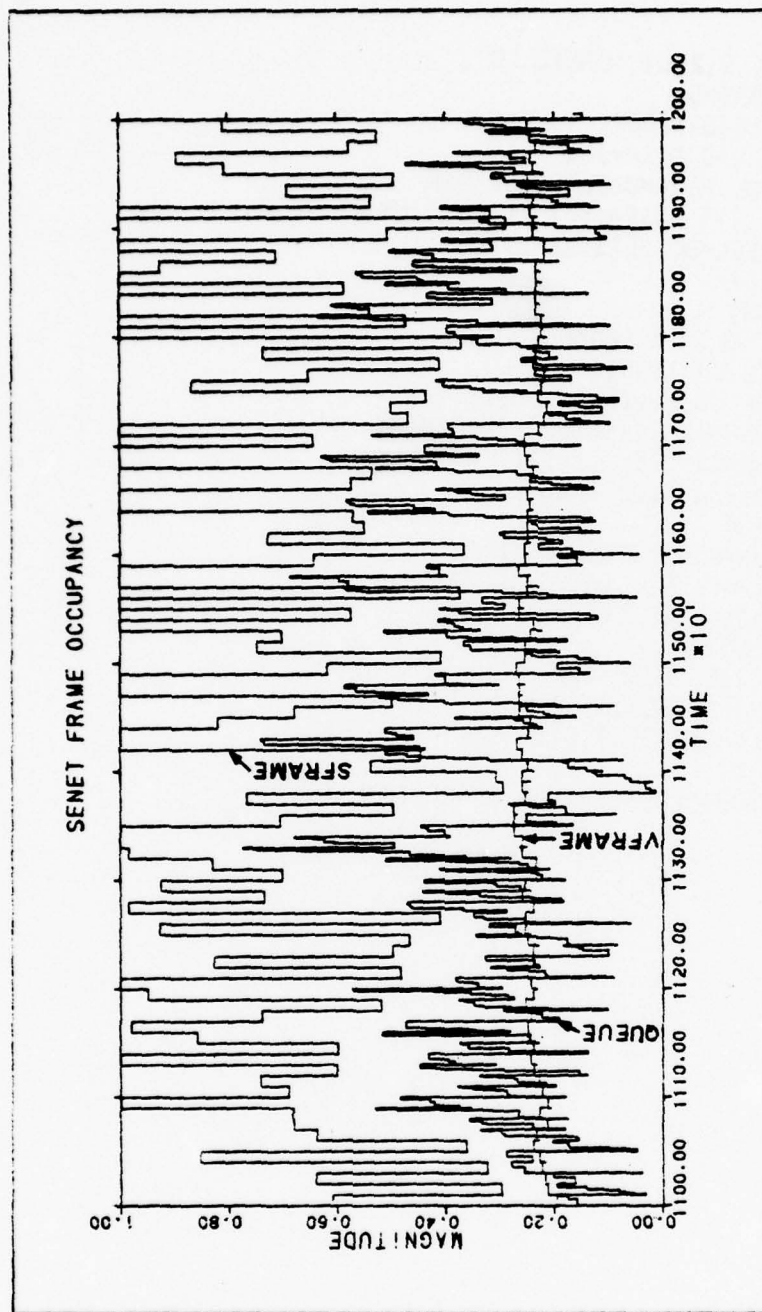
//R3363STA JOB (A100,,120,9,180),'CASTELLO',
//  NOTIFY=R3363,MSGCLASS=Q
//STAT2 EXEC PGM=SOLPLOT,REGION=180K
//STEPLIB DD DSN=R3363.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=(A,N)
//NAMME DD DSN=R3363.SOL.NAMES(MODEL),DISP=SHR
//LOG DD DSN=UR3363.LOG2,DISP=SHR,UNIT=TAPE62,
//  LABEL=1,VOL=SER=002893
//PLIDUMP DD DUMMY
//FT06F001 DD SYSOUT=(A,U)
//FT10F001 DD DSN=U1JCA.P3363.PLOT,DISP=(NEW,KEEP),
//  UNIT=TAPE9,LABEL=(1,NL),VOL=SER=001549,
//  DCB=(RECFM=VS,LRECL=364,BLKSIZE=368,DEN=2)
//SYSIN DD *
SENET FRAME OCCUPANCY
5000 15000 1000
TQUEUE 10000
TVFRAME 15440
TSFRAME 15440
TNVOICE 193
/*

```

Figure 15. JCL For Plot Program

similar to those used in the statistics program with the addition of a first card for the plot title. The variables are normalized before plotting. Therefore, the maximum value input scales the tabulated values to a realistic range.

The JCL for translating and compiling a model is given in Figure 17. The source is stored in dataset R3363.SOL.DATA member MODEL. The translated output is stored in dataset R3363.SOL.PLI in a member of the same name. This output is then compiled and saved in dataset R3363.OBJ. Finally, this dataset is linked and the load module is stored in R3363.LOAD(MODEL).



NORMALIZED
 QUEUE 40,000 bits
 VFRAME 15,440 bits
 SFRAME 15,440 bits

Figure 16. Plotted Results

```

//R3363SOL JOB (A100,,163,9,200),'CASTELLO',
//  NOTIFY=R3363,MSGCLASS=Q
//TRN EXEC PGM=CASTRAN,REGION=200K
//STEPLIB DD DSN=UR1591.LOAD,DISP=SHR
//SOLTRAN DD DSN=R3363.SOL.PLI(MODEL),DISP=SHR
//OUTF1 DD SPACE=(CYL,(1,1)),DISP=(NEW,DELETE),UNIT=SYSDA,
//  DCB=(RECFM=FB,LRECL=80,BLKSIZ=3120)
//SYSPRINT DD SYSOUT=(A,U)
//TSOOUT DD SYSOUT=(A,U)
//SYSIN DD DSN=R3363.SOL.DATA(MODEL),DISP=SHR
//PLI EXEC PGM=IELOAA,REGION=180K,
//  PARM='M,OBJECT,NODECK',COND=(EVEN,(0,LT,TRN))
//SYSUT1 DD UNIT=SYSDA,DCB=BLKSIZ=1024,SPACE=(CYL,(1,1))
//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLIN DD DISP=SHR,DSN=R3363.OBJ(MODEL)
//SYSPRINT DD SYSOUT=(A,U)
//SYSIN DD DISP=SHR,DSN=R3363.SOL.PLI(MODEL)
//SOLINC DD DISP=SHR,DSN=R1591.CASTLIB.PLI
//LKD EXEC PGM=IEWL,REGION=120K,COND=(EVEN),
//  PARM='SIZE=(114688,26624)'
//SYSLIB DD DSN=SYS1.PLIBASE,DISP=SHR
//SYSLIN DD DSN=R3363.OBJ(MODEL),DISP=SHR
//SYSLMOD DD DSN=R3363.LOAD(MODEL),DISP=SHR
//SYSPRINT DD SYSOUT=(A,U)
//SYSUT1 DD UNIT=SYSDA,DCB=BLKSIZ=1024,SPACE=(CYL,(1,1))

```

Figure 17. JCL Translate, Compile, and Link

REFERENCES

- [1] Coviello, G. J. and P. A. Vena, "Integration of Circuit/Packet Switching by a SENET (Slotted Envelope Network) Concept," National Telecommunications Conference, New Orleans, 1975.
- [2] C. G. Guffee and H. E. Ulfers, "SOL-370," Proc. of the 1975 Summer Computer Simulation Conference (Jul 1975) pp 1-11.
- [3] Knuth, D. E. and J. L. McNeley, "A Formal Definition of SOL," IEEE Trans. on Electronic Computers, EC-13, No. 5 (Aug 1964).
- [4] Norwine, A. C. and O. J. Murphy, "Characteristic Time Intervals in Telephonic Conversation," Bell System Technical Journal (1938) pp 281-291.
- [5] Brady, Paul T., "A Statistical Analysis of On-Off Patterns in 16 Conversations," Bell System Technical Journal, 47, No. 1, (Jan 1968) pp 73-91.
- [6] Horton, Jr., A. W., "The Occurrence and Effect of Lockout Occasioned by Two Echo Suppressors," BSTJ 47, No. 4 (April 1938).
- [7] Brady, Paul T., "A Technique for Investigating On-Off Patterns of Speech," Bell System Technical Journal, 44, No. 1 (January 1965) pp 1-22.
- [8] Sherman, D. N., "Storage and Delay Estimates for Asynchronous Multiplexing of Data in Speech," IEE Trans. on Communications, COM-19, No. 4 (August 1971) pp 551-555.
- [9] Jaffe, J., L. Cassotta, and S. Feldstein, "Markovian Model of Time Patterns of Speech," Science, 144 (May 15, 1964) pp 884-886.
- [10] Gustafson, H. W., "Model for the Analysis of Talkspurt and Silence Durations in Conversational Interaction," Proc. 77th Annual Conv. Amer. Psychological Assn., 44, Part I (1969) pp 43-44.
- [11] Birdsall, J. G., M. P. Ristenbatt, and S. B. Weinstein, "Analysis of Asynchronous Time Multiplexing of Speech Signals," IRE Transactions on Communications Systems (Dec 1962) pp 390-397.
- [12] Brady, Paul T., "A Model for Generating On-Off Speech Patterns in Two-way Conversation," BSTJ, 48, No. 9 (1969) pp 2445-2472.

APPENDIX A

SPEECH ACTIVITY MODEL DEVELOPMENT

This appendix discusses the development of the speech activity model used in this paper. The appendix begins with a short discussion of the speech process and previous authors' experiments and models of speech. Then the development of the speech model used in this report is traced.

Speech is not an orderly process. A speaker utters sound for a short period of time and then pauses before continuing to talk again. Some of these pauses are so short that the listener does not notice the discontinuity. A "talkspurt" is speech by a person which is discerned by the listener as continuous speech. Talkspurts are terminated when the speaker drops silent. These relatively long periods of silence between talkspurts are defined as "silent periods."

In addition to silent periods and talkspurts, the short silences within a talkspurt are of interest. These short silences are caused by stop consonants, word breaks, and other interruptions which are called "gaps" in this report. These definitions remain the same for conversational speech, but the generating mechanisms change.

Conversations consist of interactive speech between two or more people. The speech is interactive in that speakers interrupt one another and alternate talking. Interruptions are either short monosyllables, such as "yes," "right," etc... or takeover statements. Takeover statements force the initial speaker to become silent while

the interrupting speaker continues talking. Polite conversation also occurs in which a second speaker waits until the first is finished and then begins to speak. These interactions are the difference between monologue speech and conversational speech.

Some of the differences between conversational speech and monologue speech will not impact model development; others will. In conversation, talkspurts are the same as in monologue speech, but they may terminate because of intervention by the other speaker. The gaps within a talkspurt are not impacted because they occur due to the characteristics of the language and the physical speech process. The silent periods will be modified because a speaker will now pause for the same reasons as in monologue speech, plus waiting for the other speaker to pause. This change to silent periods also influences the arrival rate of talkspurts. Conversational speech also increases the number of short monosyllabic replies. We need to keep these differences in mind as we consider speech models.

Measurements have been made of both monologue and conversational speech. In this report we are concerned mainly with measurements of conversational speech. The best documented experiments are those done by Norwine and Murphy [4] and Brady [5]. While other authors have hinted at experimental work, the results of the experimentation have not been published directly.

Norwine and Murphy's experiment was done in the late 1930's. Due to the type of equipment available to them and the lack of a detailed technical description of the speech power levels involved, a comparison

with the later work of Brady is difficult. Norwine and Murphy performed their experiments to enable Horton to apply probability theory to a study of the occurrence of lockouts caused by echo suppressors and similar equipments on long distance telephone circuits [6]. Because Brady's measurements indicate that the statistics of speech events change with detector characteristics, the work of Norwine and Murphy is only of historical significance.

Brady published a series of papers in the Bell System Technical Journal; the first two described his experimental setup for measuring speech activity [7] and the results of measurements of 16 conversations [5]. His reason for studying speech activity is similar to Norwine and Murphy's - to learn more about conversational speech patterns prompted by the introduction of long telephone lines with significant delay and the increased number of voice-actuated devices. Brady's equipment detects speech by rectifying the incoming audio and comparing the rectified signal to a reference level. Each time the input signal exceeds the reference level a flip-flop is set. A clock signal resets the flip-flop every 5 ms. The output of the flip-flop is then recorded on magnetic tape, from which the signal is then digitally processed. Note that with this equipment any signal exceeding the threshold, no matter how short or when it occurs within the 5 ms interval, sets the flip-flop for the entire sample duration. Digital processing of the magnetic tape created by this sampling technique removes speech detected (active) periods less than or equal

to 15 ms and speech silence periods less than or equal to 200 ms. Therefore, the speech experiment does not consider either short speech active intervals or gaps in talkspurts.

Using this procedure Brady discovered that speech characteristics are threshold level sensitive. Some of the results of his measurements are tabulated in Table A-I. Brady's measurements, however, do not give us insight into short gaps in speech.

TABLE A-I. MEASUREMENTS OF SPEECH ACTIVITY

	Detector level in dBmØ			27.8 minutes test period
	-45	-40	-35	
Talkspurts				
Percent	43.53	39.50	35.00	
Mean	1.366	1.197	0.98	
Number	5486	5794	6224	
Mutual silence (both speakers silent)				
Percent	18.97	25.01	32.55	
Mean	1.802	1.845	1.742	
Number	5485	5792	6424	

Sherman in a concise paper to the IEEE Transactions on Communications mentions work he performed with Brady which measures the gaps in talkspurts [8]. A talkspurt is broken into active periods with mean 50 ms duration and gaps with mean 10 ms.

Stochastic models for the speech process developed by various authors give us insight into the development of a model for use in this paper. Most models of speech that this author has found are stochastic models in continuous time for a single speaker or a single conversation. Models also vary in the number of states used to

represent the speech process. The simplest model for monologue speech consists of two states and the most complex model for conversational speech consists of six states.

Models of speech activity have been developed by many authors. In the area of psychology, Jaffe, Cassotta, and Feldstein [9] developed a simple two state model with exponentially distributed hold times, and Gustafson [10] developed a two state model with modified geometric distributions for speech activity. In the case of a hundred simultaneous voice calls, Birdsall, Ristenbalt, and Weinstein approximated talkspurt arrivals with a Poisson model [11]. Brady [12] developed a six-state model for conversational speech, and Sherman [8] incorporated gaps within talkspurts in his model for monologue speech. The models developed by the last two authors are described in more detail in the following paragraphs.

Brady's six state model with is the most ambitious model of speech activity. The development of this model is based on his experimental measurements described above in Table A-I. Brady's model simulates a conversation between two talkers from talker A's viewpoint and is based on his previous experimental measurements [5]. If the model is split in half horizontally, the top half represents A talking and the bottom half represents A silent. Split the model vertically and B is talking on the right half and silent on the left. Vertical transitions are determined by talker A, and horizontal by B. Associated with each transition path between states is a parameter, q_{ij} , of a Poisson process.

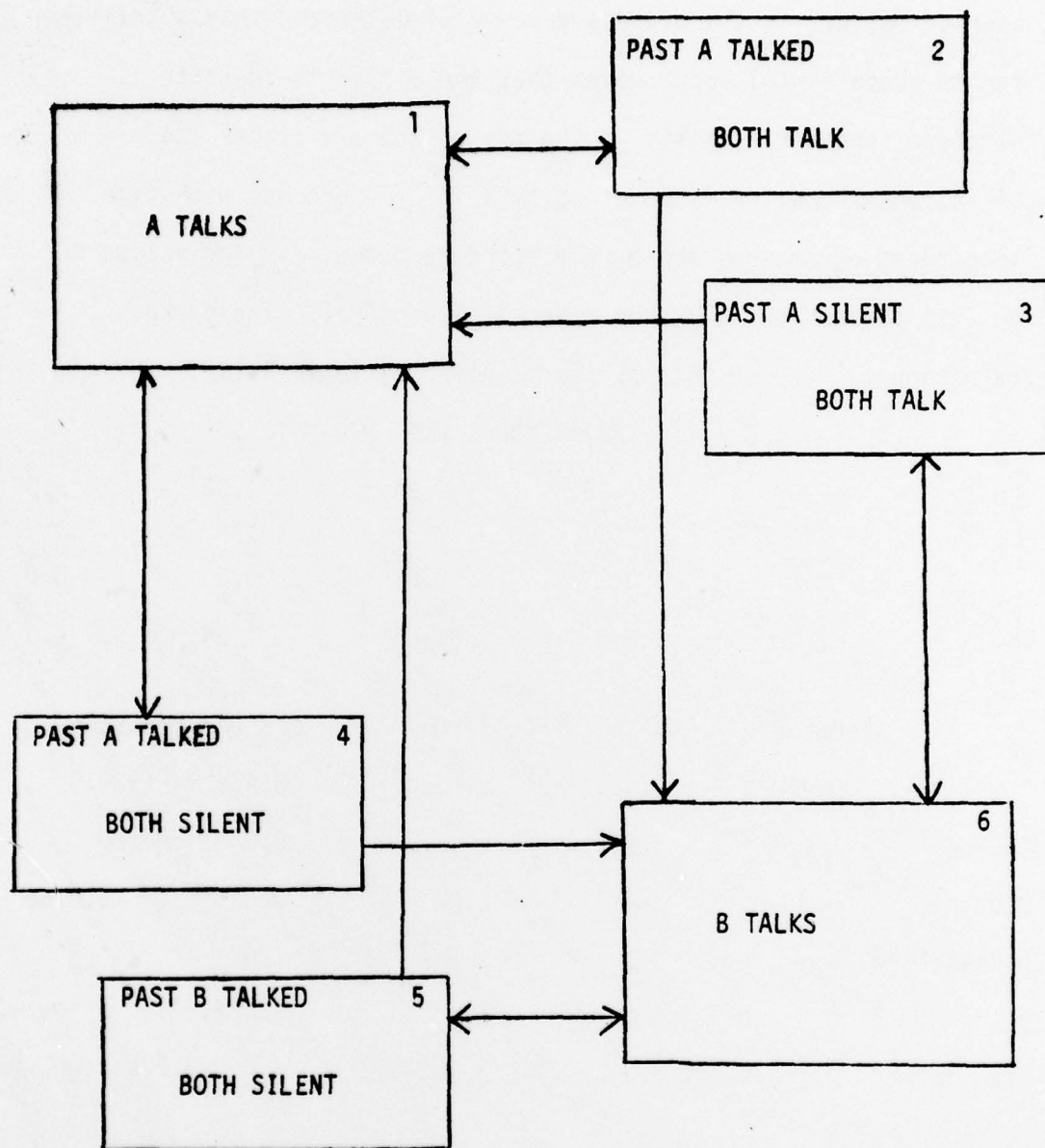


Figure A-1. BRADY MODEL

A "pulse" from the Poisson process terminates the state. Stated in other terms, q_{ij} is the average rate in times/second that a transition to state i will occur given that the system is in state j . Therefore, speaker A exists in the one of the six states and his talk-silence behavior is governed by a Poisson process with time independent parameters and by the state he occupies. The values for the exit parameters, q_{ij} , are given in Table A-II. The Markov transition matrix for this system is given by Table A-III.

TABLE A-II. BRADY MODEL EXIT PARAMETERS

0.0	2.07	2.15	2.24	1.03	0.0
0.3	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.3
0.79	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.79
0.0	2.15	2.07	1.03	2.24	0.0

TABLE A-III. MARKOV TRANSITION MATRIX FOR BRADY MODEL

$1-e^{-1.09t}$	$0.49e^{-4.22t}$	$0.51e^{-4.22t}$	$0.53e^{-3.27t}$	$0.47e^{-3.27t}$	0.0
$0.275e^{-1.09t}$	$1-e^{-4.22t}$	0.0	0.0	0.0	0.0
0.0	0.0	$1-e^{-4.22t}$	0.0	0.0	$0.275e^{-1.09t}$
$0.725e^{-1.09t}$	0.0	0.0	$1-e^{-3.27t}$	0.0	0.0
0.0	0.0	0.0	0.0	$1-e^{-3.27t}$	$0.725e^{-1.09t}$
0.0	$0.51e^{-4.22t}$	$0.49e^{-4.22t}$	$0.47e^{-3.27t}$	$0.53e^{-3.27t}$	$1-e^{-1.09t}$

Comparing the above model with his experimental work, Brady presented the following conclusions:

1. Speech activity is dependent on the characteristics of the speech detector.
2. Talkspurts tend to have an exponential distribution.

3. Silent periods cannot be accurately modeled with a simple exponential distribution.

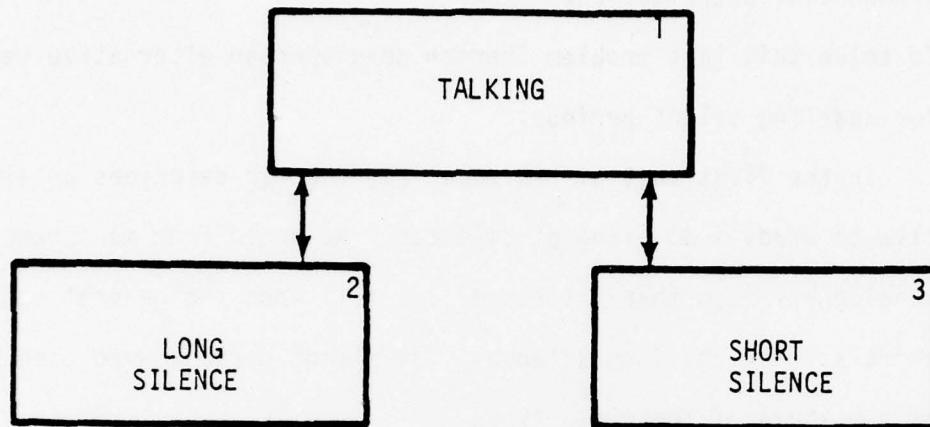
To solve this last problem Sherman developed an alternative technique for modeling silent periods.

In the first part of his paper [8] Sherman describes an alternative to Brady's modeling of silences. He noted from measurements of analogue speech that silent periods fall into two general categories: short silence and long silence. The silent periods were then modeled as a mixture of these two types.

In Sherman's model of speech activity a talker may exist in one of three states: talking, long pause or short pause. The short pause models the silent periods between words and letters within a talkspurt and the long pauses represent the silent periods between sentences. This model allows us to see the "fine" structure of talkspurts (See next page). The transition matrix for this model may be written in terms of parameters, q_{ij} , as in the Brady model above. This results in the matrix in Table A-IV, and the steady state probabilities shown in Table A-V.

TABLE A-IV. SHERMAN MODEL EXIT PARAMETERS

0.0	1.0	100.00
1.0	0.0	0.0
19.0	0.0	0.0



MONOLOGUE SPEECH MODEL
by SHERMAN

SHERMAN MODEL. Figure A-2

TABLE A-V. STEADY STATE PROBABILITIES
FOR SHERMAN MODEL

STATE	PROBABILITY
1	.4566
2	.4566
3	.0868

The Markov transition matrix for this system is given by Table A-VI.

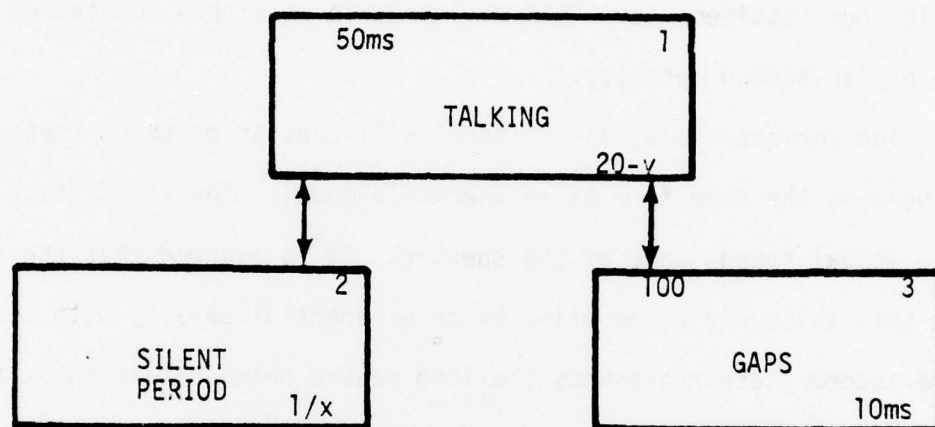
TABLE A-VI. MARKOV TRANSITION MATRIX FOR SHERMAN MODEL

$1-e^{-20t}$	e^{-t}	e^{-100t}
$0.05e^{-20t}$	$1-e^{-t}$	0.0
$0.95e^{-20t}$	0.0	$1-e^{-100t}$

This section develops a voice activity model for use in the analysis of SENET. As mentioned previously, SENET handles each direction of a full duplex circuit independently. Therefore, only one speaker need be modelled. The monologue speech activity model of Sherman modified for the statistics of conversational speech well suits our requirements. This will provide us with a continuous time model for speech activity.

The conversational speech model will consist of three states arranged in the same form as in Sherman's model. The first state models the actual sounds made by the speaker. It is assumed that the duration in this state may be modelled by an exponential density with mean 50 ms. The second state represents the long pauses between talking states due to pauses between thoughts and pauses while the other speaker is talking. The duration in this state is assumed to also have an exponential density with mean $1/x$. The third state which represents the short silences within a talkspurt remains unchanged from Sherman's model.

The most significant changes from Sherman's model is in the duration of the silent period and the rates at which states two and three are entered from state one. The assumption that state one has an exponential duration with mean 50 ms implies that the exit rate from this state will be 20 times/second. If we let y represent the rate of exit to state two, then state one must exit to state three at a rate of $20-y$. From the assumptions made for states two and three they will have exit rates to state one of x and 100 respectively. These are shown on Figure A-3.



PRELIMINARY SPEECH MODEL. Figure A-3

Based on the results of Brady's experimental measurements values may be determined for x and y . Brady has indicated that the measurement made at -45 dBm \emptyset are too cluttered with noise and those made at -35 dBm \emptyset clip talkspurts. Therefore, we will use the statistics of voice measured at -40 dBm \emptyset . Brady filtered out the very short talkspurts and bridged the short silences. Therefore, his "talkspurt" encompasses what we consider as two separate states: states one and three.

If we consider states one and three as a single talkspurt state then x is the rate of talkspurt arrivals. Based on Brady's measurements on conversation speech made at -40 dBm \emptyset there were 5794 talkspurts during the 274.8 minutes of conversation speech. This results in an arrival rate in talkspurts per second or an exit rate from long silent periods of

$$x = \frac{5794}{60 \times 274.8} = 0.3514.$$

A value for y may be determined from Brady's measured probability of being in the silent state. Again using the -40 dBm \emptyset measurements the probability of being in state two is 0.605. We assume that this is the steady state probability of being in our state two. The following method may be used to determine a value for y . Let p_i be the steady state probability of being in state i . The following equations hold for steady state.

$$yp_i = 0.3514 p_2$$

$$(20-y)p_1 = 100p_3$$

$$1 = p_1 + p_2 + p_3$$

$$p_2 = 0.605$$

In matrix notation this may be written as

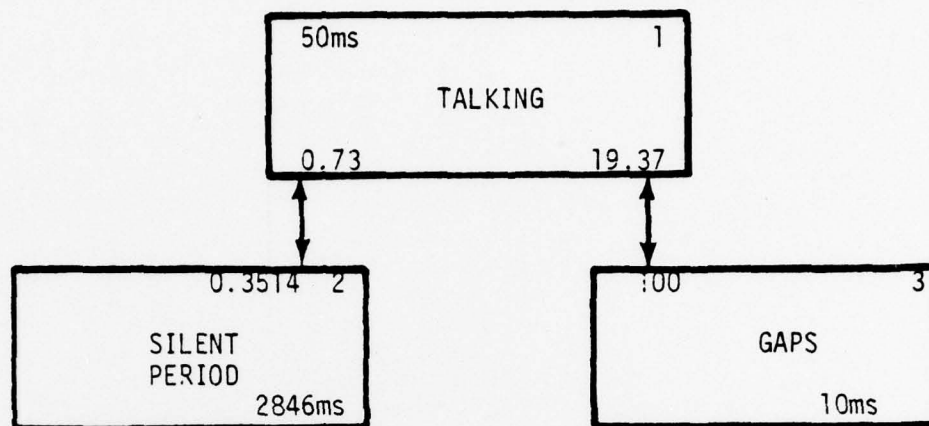
$$\begin{bmatrix} y & 0.3514 & 0 \\ 20-y & 0 & 100 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ 0.605 \\ p_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Solving results in $y = 19.37$.

With these values for the unknowns the final continuous time model is given in Figure A-4. The steady state probabilities are given in table A-VII, the transition matrix is given in Table A-VII, and the continuous model exit parameters are given in Table A-IX.

TABLE A-VII. STEADY STATE PROBABILITIES
FOR CONTINUOUS MODEL

STATE	PROBABILITY
1	0.3309
2	0.6050
3	0.0614



SPEECH MODEL. Figure A-4

TABLE A-VIII. MARKOV TRANSITION MATRIX FOR CONTINUOUS MODEL

$1 - e^{-20t}$	$e^{-0.3514t}$	e^{-100t}
$0.0315e^{-20t}$	$1 - e^{-0.3514t}$	0.0
$0.9685e^{-20t}$	0.0	$1 - e^{-100t}$

TABLE A-IX. CONTINUOUS MODEL EXIT PARAMETERS

0.0	0.3514	100.00
1.63	0.0	0.0
19.37	0.0	0.0

DISTRIBUTION LIST

STANDARD:

R100 - 2	R200 - 1
R102/R103/R103R - 1	R300 - 1
R102M - 1	R400 - 1
R102T - 9 (8 for stock)	R500 - 1
R104 - 1	R700 - 1
R110 - 1	R800 - 1
R123 - 1 (Library)	NCS-TS - 1
R124A - 1 (for Archives)	

205 - 13 (Unclassified/Unlimited Distribution)

DCA-EUR - 1 (Defense Communications Agency European Area
ATTN: Technical Director
APO New York 091311)

DCA-PAC - 1 (Defense Communications Agency Pacific Area
ATTN: Technical Director
Wheeler AFB, HI 96854)

USDCFO - 1 (Chief, USDCFO/US NATO
APO New York 09667)